METHOD AND APPARATUS FOR CONSTRUCTING A SPEECH FILTER USING ESTIMATES OF CLEAN SPEECH AND NOISE

BACKGROUND OF THE INVENTION

The present invention relates to speech processing. In particular, the present invention relates to speech enhancement.

In speech recognition, it is common to enhance the speech signal by removing noise before performing speech recognition. Under some systems, this is done by estimating the noise in the speech signal and subtracting the noise from the noisy speech signal. This technique is typically referred to as spectral subtraction because it is performed in the spectral domain.

Since it is impossible to estimate the noise in a speech signal perfectly, any estimate that is used in spectral subtraction will have some amount of error. Because of this error, it is possible that the estimate of the noise in the noisy speech signal will be larger than the noisy speech signal for some frames of the signal. This would produce a negative value for the "clean" speech, which is physically impossible.

To avoid this, spectral subtraction systems rely on a set of parameters that are set by hand to allow for maximum noise reduction while ensuring a stable system. Relying on such parameters is undesirable since they are typically noise-source

dependent and thus must be hand-tuned for each type of noise-source.

Other systems attempt to enhance the speech signal using a Weiner filter to filter out the noise in the speech signal. In such systems, the gain of the Weiner filter is generally based on a signal-to-noise ratio. To arrive at the proper gain value, the level of the noise in the signal must be determined.

One common technique for determining the level of noise is to estimate the noise during non-speech segments in the speech signal. This technique is less than desirable because it not only requires a correct estimate of the noise during the non-speech segments, it also requires that the non-speech segments be properly identified as not containing speech. In addition, this technique depends on the noise being stationary (non-changing). If the noise is changing over time, the estimate of the noise will be wrong and the filter will not perform properly.

Another system for enhancing speech attempts to identify a clean speech signal using a probabilistic framework that provides a Minimum Mean Square Error (MMSE) estimate of the clean signal given a noisy speech signal. Unfortunately, such systems can provide poor estimates of the clean speech signal at times, especially when the signal-to-noise ratio is low. As a result, using the clean speech estimates directly in speech recognition can result in poor recognition accuracy.

Thus, a system is needed that does not require as much hand-tuning of parameters as in spectral subtraction while avoiding the poor estimates that sometimes occur in MMSE estimation.

SUMMARY OF THE INVENTION

A method and apparatus identify a clean speech signal from a noisy speech signal. To do this, a clean speech value and a noise value are estimated from the noisy speech signal. The clean speech value and the noise value are then used to define a gain on a filter. The noisy speech signal is applied to the filter to produce the clean speech signal. Under some embodiments, the noise value and the clean speech value are used in both the numerator and the denominator of the filter gain, with the numerator being guaranteed to be positive.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram of a general computing environment in which the present invention may be practiced.

FIG. 2 is a block diagram of a mobile device in which the present invention may be practiced.

FIG. 3 is a block diagram of a speech enhancement system under one embodiment of the present invention.

FIG. 4 is a flow diagram of a speech enhancement method under one embodiment of the present invention.

FIG. 5 is a flow diagram of a simplified method for determining clean speech and noise estimates under one embodiment of the present invention.

DETAILED DESCRIPTION OF ILLUSTRATIVE EMBODIMENTS

FIG. 1 illustrates an example of a suitable computing system environment 100 on which the invention may be implemented. The computing system environment 100 is only one example of a suitable computing environment and is not intended to suggest any limitation as to the scope of use or functionality of the invention. Neither should the computing environment 100 be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in the exemplary operating environment 100.

The invention is operational with numerous other general purpose or special purpose computing system environments or configurations. Examples of well-known computing systems, environments, and/or configurations that may be suitable for use with the invention include, but are not limited to, personal computers, server computers, hand-held or laptop devices, multiprocessor systems, microprocessor-based systems, set top boxes, programmable consumer electronics, network PCs, minicomputers, mainframe computers, telephony systems, distributed computing environments that include any of the above systems or devices, and the like.

The invention may be described in the general context of computer-executable instructions, such as program modules, being executed by a computer. Generally, program modules include routines, programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types. The invention is designed to be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules are located in both local and remote computer storage media including memory storage devices.

With reference to FIG. 1, an exemplary system for implementing the invention includes a general-purpose computing device in the form of a computer 110. Components of computer 110 may include, but are not limited to, a processing unit 120, a system memory 130, and a system bus 121 that couples various system components including the system memory to the processing unit 120. The system bus 121 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. By way of example, and not limitation, such architectures include Industry Standard Architecture (ISA) bus, Micro Channel Architecture (MCA) bus, Enhanced ISA (EISA) bus, Video Electronics Standards Association (VESA) local

bus, and Peripheral Component Interconnect (PCI) bus also known as Mezzanine bus.

Computer 110 typically includes a variety of computer readable media. Computer readable media can be any available media that can be accessed by computer 110 and includes both volatile and nonvolatile media, removable and non-removable media. By way of example, and not limitation, computer readable media may comprise computer storage media and communication media. Computer storage media includes both volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information such as computer readable instructions, data structures, program modules or other data. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical disk storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can be accessed by computer 110. Communication media typically embodies computer readable instructions, data structures, program modules or other data in a modulated data signal such as a carrier wave or other transport mechanism and includes any information delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to

encode information in the signal. By way of example, and not limitation, communication media includes wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared and other wireless media. Combinations of any of the above should also be included within the scope of computer readable media.

The system memory 130 includes computer storage media in the form of volatile and/or nonvolatile memory such as read only memory (ROM) 131 and random access memory (RAM) 132. A basic input/output system 133 (BIOS), containing the basic routines that help to transfer information between elements within computer 110, such as during start-up, is typically stored in ROM 131. RAM 132 typically contains data and/or program modules that are immediately accessible to and/or presently being operated on by processing unit 120. By way of example, and not limitation, FIG. 1 illustrates operating system 134, application programs 135, other program modules 136, and program data 137.

The computer 110 may also include other removable/non-removable volatile/nonvolatile computer storage media. By way of example only, FIG. 1 illustrates a hard disk drive 141 that reads from or writes to non-removable, nonvolatile magnetic media, a magnetic disk drive 151 that reads from or writes to a removable, nonvolatile magnetic disk 152, and an optical disk drive 155 that reads from or writes to a removable, nonvolatile optical disk 156 such as a CD

ROM or other optical media. Other removable/non-removable, volatile/nonvolatile computer storage media that can be used in the exemplary operating environment include, but are not limited to, magnetic tape cassettes, flash memory cards, digital versatile disks, digital video tape, solid state RAM, solid state ROM, and the like. The hard disk drive 141 is typically connected to the system bus 121 through a non-removable memory interface such as interface 140, and magnetic disk drive 151 and optical disk drive 155 are typically connected to the system bus 121 by a removable memory interface, such as interface 150.

The drives and their associated computer storage media discussed above and illustrated in FIG. 1, provide storage of computer readable instructions, data structures, program modules and other data for the computer 110. In FIG. 1, for example, hard disk drive 141 is illustrated as storing operating system 144, application programs 145, other program modules 146, and program data 147. Note that these components can either be the same as or different from operating system 134, application programs 135, other program modules 136, and program data 137. Operating system 144, application programs 145, other program modules 146, and program data 147 are given different numbers here to illustrate that, at a minimum, they are different copies.

A user may enter commands and information into the computer 110 through input devices such as a keyboard 162, a microphone 163, and a pointing device

161, such as a mouse, trackball or touch pad. Other input devices (not shown) may include a joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 120 through a user input interface 160 that is coupled to the system bus, but may be connected by other interface and bus structures, such as a parallel port, game port or a universal serial bus (USB). A monitor 191 or other type of display device is also connected to the system bus 121 via an interface, such as a video interface 190. In addition to the monitor, computers may also include other peripheral output devices such as speakers 197 and printer 196, which may be connected through an output peripheral interface 195.

The computer 110 is operated in a networked environment using logical connections to one or more remote computers, such as a remote computer 180. The remote computer 180 may be a personal computer, a hand-held device, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to the computer 110. The logical connections depicted in FIG. 1 include a local area network (LAN) 171 and a wide area network (WAN) 173, but may also include other networks. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets and the Internet.

When used in a LAN networking environment, the computer 110 is connected to the LAN 171 through a network interface or adapter 170. When used in a WAN networking environment, the computer 110 typically includes a modem 172 or other means for establishing communications over the WAN 173, such as the Internet. The modem 172, which may be internal or external, may be connected to the system bus 121 via the user input interface 160, or other appropriate mechanism. In a networked environment, program modules depicted relative to the computer 110, or portions thereof, may be stored in the remote memory storage device. By way of example, and not limitation, FIG. 1 illustrates remote application programs 185 as residing on remote computer 180. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

FIG. 2 is a block diagram of a mobile device 200, which is an exemplary computing environment. Mobile device 200 includes a microprocessor 202, memory 204, input/output (I/O) components 206, and a communication interface 208 for communicating with remote computers or other mobile devices. In one embodiment, the afore-mentioned components are coupled for communication with one another over a suitable bus 210.

Memory 204 is implemented as non-volatile electronic memory such as random access memory (RAM)

with a battery back-up module (not shown) such that information stored in memory 204 is not lost when the general power to mobile device 200 is shut down. A portion of memory 204 is preferably allocated as addressable memory for program execution, while another portion of memory 204 is preferably used for storage, such as to simulate storage on a disk drive.

Memory 204 includes an operating system 212, application programs 214 as well as an object store 216. During operation, operating system 212 is preferably executed by processor 202 from memory 204. Operating system 212, in one preferred embodiment, is a WINDOWS® CE brand operating system commercially available from Microsoft Corporation. Operating system 212 is preferably designed for mobile devices, and implements database features that can be utilized by applications 214 through a set of exposed application programming interfaces and methods. The objects in object store 216 are maintained by applications 214 and operating system 212, at least partially in response to calls to the exposed application programming interfaces and methods.

Communication interface 208 represents numerous devices and technologies that allow mobile device 200 to send and receive information. The devices include wired and wireless modems, satellite receivers and broadcast tuners to name a few. Mobile device 200 can also be directly connected to a computer to exchange data therewith. In such cases, communication interface 208 can be an infrared

transceiver or a serial or parallel communication connection, all of which are capable of transmitting streaming information.

Input/output components 206 include a variety of input devices such as a touch-sensitive screen, buttons, rollers, and a microphone as well as a variety of output devices including an audio generator, a vibrating device, and a display. The devices listed above are by way of example and need not all be present on mobile device 200. In addition, other input/output devices may be attached to or found with mobile device 200 within the scope of the present invention.

The present invention provides a method and apparatus for enhancing a speech signal. FIG. 3 provides a block diagram of the system and FIG. 4 provides a flow diagram of the method of the present invention.

At step 400, a noisy analog signal 300 is converted into a sequence of digital values that are grouped into frames by a frame constructor 302. Under one embodiment, the frames are constructed by applying analysis windows to the digital values where each analysis window is a 25 millisecond hamming window, and the centers of the windows are spaced 10 milliseconds apart.

At step 402, a frame of the digital speech signal is provided to a Fast Fourier Transform 304 to compute the phase and magnitude of a set of frequencies found in the frame. The magnitude or the

square of the magnitude of each FFT is then selected/determined by block 305 at step 403.

At step 404, the magnitude values are optionally applied to a Mel-scale filter bank 306, which applies perceptual weighting to the frequency distribution and reduces the number frequency bins that are associated with the frame. The Mel-scale filter bank is an example of a frequency-based transform. In such transforms, the level of filtering applied to a frequency is based on the identity of the frequency or the magnitudes of the frequencies are scaled and combined to form fewer parameters. Thus, in FIG. 3, if the frequency values are not applied to the Mel-scale filter bank, they are not applied to a frequency-based transform.

A log function 310 is applied to the values from magnitude block 305 or Mel-Scale filter bank 306 (if the filter bank is used) at step 408 to compute the logarithm of each frequency magnitude.

At step 410, the logarithms of each frequency are applied to a discrete cosine transform (DCT) 312 to form a set of values that are represented as an observation feature vector. If the Mel-scale filter bank was used, the observation vector is referred to as a Mel-Frequency Cepstral Coefficient (MFCC) vector. If the Mel-scale filter bank was not used, the observation vector is referred to as a High Resolution Cepstral Coefficient (HRCC) vector, since it retains all of the frequency information from the input signal.

The observation feature vector is applied to a maximum likelihood (ML) estimation block 314 at step 412. ML estimation block 314 builds a maximum likelihood estimation of a noise model based on a sequence of observation feature vectors that represent an utterance, typically a sentence. Under one embodiment, this noise model is a single Gaussian distribution that is described by its mean and covariance.

The noise model and the observation feature vectors are provided to a clean speech and noise estimator 316 together with parameters 315 that describe a prior clean speech model. Under one embodiment the prior clean speech model is a Gaussian Mixture Model that is defined by a mixture weight, a mean, and a covariance for each of a set of mixture components. Using the model parameters for the clean speech and the noise, estimator 316 generates an estimate of a clean speech value and a noise value for each frame of the input speech signal at step 414. Under one embodiment, the estimates are Minimum Mean Square Error (MMSE) estimates that are computed as:

$$\hat{x}_t = \int x p(x \mid y_t, \Lambda_x, \Lambda_n) dx \qquad \text{EQ. 1}$$

$$\hat{n}_t = \int n p(n \mid y_t, \Lambda_x, \Lambda_n) dn \qquad \text{EQ. 2}$$

where $\hat{x}_t$ is the MMSE estimate of the clean speech, $\hat{n}_t$ is the MMSE estimate of the noise, x is a clean speech value, n is a noise value, y_t is the

observation feature vector, Λ_n represents the parameters of the noise model, and Λ_x represents the parameters of the clean speech model.

At steps 416, the clean speech estimate and the noise estimate, which are in the cepstral domain, are applied to an inverse discrete cosine transform 317. The results of the inverse discrete cosine transform are applied to an exponential function 318 at step 418. This produces spectral values for the clean speech estimate and the noise estimate.

At step 420, the spectral values for the clean speech estimate and the noise estimate are smoothed over time and frequency by a smoothing block 322. The smoothing over time involves smoothing each frequency value in the spectral values across different frames of the speech signal. Under one embodiment, the smoothing over frequency involves averaging values of neighboring frequency bins within a frame and placing the average value at a frequency position that is in the center of the frequency bins used to form the average value.

The smoothed spectral values for the estimate of the clean speech signal and the estimate of the noise are then used to determine the gain for a Weiner filter 326 at step 422. Under one embodiment, the gain of the Weiner filter is set as:

$$|H(t,f)| = \frac{|\hat{P}_x(t,f)|^2 + (1-\alpha)|\hat{P}_n(t,f)|^2}{|\hat{P}_x(t,f)|^2 + |\hat{P}_n(t,f)|^2} \qquad \text{EQ. 3}$$

where $|H(t,f)|$ is the gain of the Weiner filter, $|\hat{P}_x(t,f)|^2$ is the power spectrum of the clean speech estimate, $|\hat{P}_n(t,f)|^2$ is the power spectrum of the noise estimate, and α is factor that avoids over estimation of the noise spectra. Values for α vary from .6 to .95 according to the local SNR computed from the ratio of $|\hat{P}_x(t,f)|^2$ to $|\hat{P}_n(t,f)|^2$. t and f are time and frequency indices, respectively. If the Mel-Scale filter bank was used, f is the indices of the filter bank.

In Equation 3, actual estimates of the noise and clean speech are used in the denominator. In addition, the estimate of the noise in the numerator is multiplied by the factor $1-\alpha$ such that the product is always guaranteed to be positive. This ensures that the gain will be positive regardless of the value estimated for the noise. This makes the system of the present invention much more stable than spectral subtraction systems and does not require the setting of as many parameters as spectral subtraction.

Once the filter gain has been determined at step 422, the power spectrum of the noisy frequency domain values produced by magnitude block 305 or Mel-Scale filter bank 306 is applied to the Weiner filter at step 424 to produce a filtered clean speech power spectrum. Specifically:

$$|\tilde{P}_x(t,f)|^2 = |P_y(t,f)|^2 \cdot |H(t,f)| \qquad \text{EQ. 4}$$

where $|H(t,f)|$ is the gain of the Weiner filter, $|\tilde{P}_x(t,f)|^2$ is the filtered clean speech power spectrum, and $|P_y(t,f)|^2$ is the power spectrum of the noisy speech signal.

At step 426, the filtered clean speech power spectrum 328 can be used to generate a clean speech signal that is to be heard by a user or it can be applied to a feature extraction unit 330, such as a Mel-Frequency Cepstral Coefficient feature extraction unit, as pre-processing for speech recognition.

JOINT MODEL FOR SPEECH AND NOISE

It is assumed that the speech and noise waveforms mix linearly in the time domain. As a result of this assumption, it is common to model the noisy cepstral features y as a first order Taylor series in x and n.

$$y = A(x_0,n_0) + G(x_0,n_0)(x-x_0) + (I - G(x_0,n_0))(n-n_0) + \varepsilon$$

EQ. 5

$$A(x,n) = C\log\left(\exp\left(C^{-1}x\right) + \exp\left(C^{-1}n\right)\right) \qquad \text{EQ. 6}$$

$$G(x,n) = C\frac{1}{\exp\left(C^{-1}(n-x)\right)+1}C^{-1} \qquad \text{EQ. 7}$$

The symbol I denotes the identity matrix. From now on, we will use the shorthand notation $A_0 = A(x_0,n_0)$ and $G_0 = G(x_0,n_0)$. In practice, it is useful to set all of the off-diagonal elements of G_0 to zero. This reduces computational requirements

drastically, while introducing a slight increase in distortion.

Assuming the residual error term ε is an independent Gaussian, this induces a Gaussian probability distribution on y given x and n.

$$p(y \mid x,n) = N(y;\mu_y,\Sigma_\varepsilon) \qquad \text{EQ. 8}$$

$$\mu_y = A_0 + G_0(x_t - x_0) + (I - G_0)(n_t - n_0) \qquad \text{EQ. 9}$$

Before using this model to enhance speech, it is necessary to add a prior model for speech, Λ_x, and a prior model for noise, Λ_n. Under one embodiment of the present invention, the prior model for speech is a Gaussian mixture morel, and the prior model for noise is a single Gaussian component:

$$p(x,i) = N(y;m_x(i),\Sigma_x(i))c_i \qquad \text{EQ. 10}$$

$$p(n) = N(y;m_n,\Sigma_n) \qquad \text{EQ. 11}$$

Finally, the joint model of noisy observation, clean speech, noise, and speech state is:

$$p(y,x,n,i \mid \Lambda_x,\Lambda_n) = p(y \mid x,n)p(x,i)p(n) \qquad \text{EQ. 12}$$

The joint model of equation 12 can be manipulated to produce several formulae useful in estimating clean speech, noise, and speech state from the noisy observation.

First, the clean speech state can be inferred as:

$$p(i \mid y) = N(y;\mu_y(i),\Sigma_y(i)) \qquad \text{EQ. 13}$$

$$\mu_y(i) = A_0 + G_0(m_x(i) - x_0) + (I - G_0)(m_n - n_0) \quad \text{EQ. 14}$$

$$\Sigma_y(i) = (I - G_0)\Sigma_n(I - G_0)' + G_0\Sigma_x G_0' + \Sigma_\varepsilon \quad \text{EQ. 15}$$

Second, the clean speech vector can be inferred as:

$$p(x \mid y,i) = N(x; \mu_{x|y}(i), \Sigma_{x|y}(i)) \quad \text{EQ. 16}$$

$$\mu_{x|y}(i) = m_x(i) + (\Sigma_y(i))^{-1} G_0 \Sigma_x(i)(y - \mu_y(i)) \quad \text{EQ. 17}$$

$$\Sigma_{x|y}(i) = (\Sigma_y(i))^{-1}((I - G_0)\Sigma_n(I - G_0)' + \Sigma_\varepsilon)\Sigma_x(i) \quad \text{EQ. 18}$$

Third, the noise vector can be inferred as:

$$p(n \mid y,i) = N(x; \mu_{n|y}(i), \Sigma_{n|y}(i)) \quad \text{EQ. 19}$$

$$\mu_{n|y}(i) = m_n + (\Sigma_y(i))^{-1}(I - G_0)\Sigma_n(y - \mu_y(i)) \quad \text{EQ. 20}$$

$$\Sigma_{n|y}(i) = (\Sigma_y(i))^{-1}(G_0\Sigma_x(i)G_0' + \Sigma_\varepsilon)\Sigma_n \quad \text{EQ. 21}$$

ML ESTIMATION OF NOISE DISTRIBUTION

Step 412, in which a Maximum Likelihood estimate of the noise distribution is determined, involves identifying parameters, Λ_n, that maximize the joint probability $P(Y,X,N,I \mid \Lambda_x,\Lambda_n)$ given y_t and Λ_x, where Y is the sequence of observation vectors, X is the sequence of clean speech vectors, N is the sequence of noise vectors, I is the sequence of mixture component indices, Λ_x represents the parameters of the clean speech model, which consist of mixture component weights c_i, mixture component means $m_x(i)$, and mixture component covariances $\Sigma_x(i)$,

and Λ_n represents the parameters of the noise model, which consist of a mean m_n and a covariance Σ_n.

Under one embodiment of the present invention, an iterative Expectation-Maximization algorithm is used to identify the parameters of the noise model. Specifically, the parameters are updated during the M-step of the EM algorithm as:

$$\hat{m}_n = \frac{\sum_t \sum_i p(i \mid y_t)\mu_{n|y}(i)}{\sum_t \sum_i p(i \mid y_t)} \qquad \text{EQ. 22}$$

$$\hat{\Sigma}_n = diag[\frac{\sum_t \sum_i p(i \mid y_t)[\mu_{n|y}(i)\mu_{n|y}(i)'+\Sigma_{n|y_t}(i)]}{\sum_t \sum_i p(i \mid y_t)} - \hat{m}_n \hat{m}_n'] \qquad \text{EQ. 23}$$

where the notation ()' indicates a transpose, t is a frame index, i is a mixture component index, $\hat{m}_n$ is the updated mean of the noise model, m_n is the past mean of the noise model, $\hat{\Sigma}_n$ is the updated covariance of the noise model, $p(i \mid y_t)$ is a posterior mixture component probability (defined in equations 13-15), and $\mu_{n|y}(i)$ and $\Sigma_{n|y_t}(i)$ are a mean and covariance for a posterior distribution, defined in equations 20 and 21.

The covariance matrix, Σ_ε, of the residue error can be derived with an iterative EM process by:

$$\hat{\Sigma}_\varepsilon = diag[\frac{\sum_t \sum_i p(i \mid y_t) E\{\varepsilon_t \varepsilon_t' \mid y_t, i\}}{\sum_t \sum_i p(i \mid y_t)}] \qquad \text{EQ. 24}$$

where $E\{\varepsilon_t\varepsilon_t' \mid y_t, i\}$ is the expectation of the residue error. Under one embodiment, this exact estimation is not adopted because it involves a large number of computations and because it requires stereo training data that includes both noisy speech and clean speech in order to collect training samples of the residue so that the expected value of the residue can be determined. Instead, the covariance is either set to zero or approximated as:

$$\hat{\Sigma}_\varepsilon = \max(0, \Sigma_\varepsilon + diag[\frac{\sum_t \sum_i p(i \mid y_t)[(y_t - \mu_y(i))(y_t - \mu_y(i))' - \Sigma_y(i)}{\sum_t \sum_i p(i \mid y_t)}])$$

EQ. 25

where the max operation ensures that the values of the matrix are non-negative. Note that equation 25 does not require stereo training data. Instead the covariance is set directly from the observation vectors.

The convergence of equations 22 and 23 becomes very slow if Σ_n is small. Under one embodiment, this is overcome by maximizing $P(Y, I \mid \Lambda_x \Lambda_n)$ instead of $P(Y, X, N, I \mid \Lambda_x \Lambda_n)$. By setting the derivative of the corresponding auxiliary function with respect to m_n to zero, the update for the mean becomes:

$$\hat{m}_n = m_n + \frac{\sum_t \sum_i p(i \mid y_t)(I - G_0)\Sigma_y^{-1}(i)(y_t - \mu_y(i))}{\sum_t \sum_i p(i \mid y_t)(I - G_0)\Sigma_y^{-1}(i)} \quad \text{EQ. 26}$$

The update for the covariance $\hat{\Sigma}_n$ remains the same as shown in Equation 23. Note that in

Equation 26, the covariance of the noise model Σ_n has been removed from the numerator, making the update converge faster if the covariance Σ_n is small.

MMSE estimation of Clean Speech and Noise

Once the noise model has been constructed, an estimate of the noise for each frame is computed as:

$$\hat{n}_t = \int np(n \mid y_t)dn = \sum_i p(i \mid y_t) \int np(n \mid y_t, i)dn = \sum_i p(i \mid y_t)\mu_{n|y}(i)$$

EQ. 27

Similarly, the estimate of the clean speech signal is computed as:

$$\hat{x}_t = \sum_i p(i \mid y_t)\mu_{x|y}(i) \qquad \text{EQ. 28}$$

Simplified Determination of Model Parameters and Estimates of Clean Speech and Noise

Under one embodiment, the ML computations and the noise and clean speech estimations described above are simplified. A flow diagram of the simplified technique is shown in FIG. 5.

At step 500 of FIG. 5, an observation vector for a frame is selected. At step 502, the posterior probability $p(i \mid y_t)$ for each mixture component i is computed. The mixture component with the highest posterior probability is then selected at step 504. Instead of using all of the mixture components in computing the noise estimate, only the selected mixture component is used.

At step 506, a variable $ddnx_0$ is initialized for the frame. This variable is defined as:

$$ddnx_0 = (n_0 - x_0(i)) - (m_n - m_x(i)) \qquad \text{EQ. 29}$$

However, it is not computed explicitly using this definition.

For the first frame, $ddnx_0$ is initialized to zero. For each subsequent frame, the initial value for $ddnx_0$ is set to the value in the past frame plus the difference between the mean of the posterior of the selected mixture component in the current frame and the mean of the posterior of the selected mixture component in the past frame. Note that different mixture components may be selected in different frames.

After $ddnx_0$ has been initialized, it is iteratively updated at steps 508 and 510 using an update equation of:

$$ddnx_0 = (\Sigma_y(i))^{-1}((I - G_0)\Sigma_n - G_0\Sigma_x(i))(y - \mu_y(i)) \qquad \text{EQ. 30}$$

After a desired number of iterations have been performed at step 510 (in one embodiment four iterations are used), the process continues at step 512 where the value for $ddnx_0$ is used to compute the clean speech and noise estimates for the frame according to the above equations, where G_0 can be computed from $ddnx_0$ according to equation 31, and equation 14 is modified according to equation 32.

$$G_0 = C\frac{1}{\exp\left(C^{-1}\left(ddnx_0 + \left(m_n - m_x(i)\right)\right)\right)+1}C^{-1}$$

EQ. 31

$$\mu_y(i) = m_x(i) + C\log\left(1+\exp\left(C^{-1}\left(ddnx_0 + \left(m_n - m_x(i)\right)\right)\right)\right) - \left(I - G_0\right)ddnx_0$$

EQ. 32

After the clean speech and noise estimates have been determined for the frame, the method determines if there are more frames to process at step 514. If there are more frames, the method returns to step 500 to select the next frame. If the last frame has been processed, the method ends after step 514.

Although the present invention has been described with reference to particular embodiments, workers skilled in the art will recognize that changes may be made in form and detail without departing from the spirit and scope of the invention.